Title:

On the Limitations of Embedding Data in Immutable Carriers

Author(s):

Gina Fisk, 115088, CCS-1
Eric Weigle, 171468, CCS-1

Submitted to:

ACM MM Workshop on Security and Multimedia

On the Limitations of Embedding Data in Immutable Carriers

Gina Fisk
Los Alamos National Laboratory
gina@lanl.gov

Eric Weigle
Los Alamos National Laboratory
ehw@lanl.gov

ABSTRACT

Null ciphers are some of the oldest cited examples of modern steganography, and are some of the few steganographic algorithms that use either synthetic or immutable carriers. In contrast, the vast majority of today's steganographic algorithms use mutable carriers where the embedding process requires modifying the carrier in some way. The main deficiency with mutating the carrier during the embedding process is that the algorithms will leave some sort of signature. In this paper we explore the idea of embedding data without changing the carrier by mapping the message onto the carrier instead of making modifications to the carrier. We explore algorithms that use *Variable Interval Symbol Aggregation (VISA)* for both text and binary data. We study these *variable interval* algorithms in terms of several quantitative measures and show that these algorithms, often cited as classic examples of steganography, share many characteristics with encryption algorithms.

1. INTRODUCTION

Steganographic transmissions are designed to be imperceptible. However, to date there have been no published steganographic implementations that do not change the carrier in some discernible fashion. While some algorithms change the images in very subtle ways [8, 10], any algorithm that modifies the carrier has the potential to be detected [1, 2, 6, 3, 4, 12]. Additionally, carriers that are digitally signed or watermarked cannot be considered as potential carriers for traditional steganographic algorithms since the embedding may disrupt the authentication techniques that are in place.

To begin to address these limitations of current steganographic techniques, we explore and analyze several algorithms that map messages onto carriers instead of embedding data into the carriers. We introduce a class of *variable interval* algorithms that have no potential to leave a signature in the carrier. Instead of modifying selected bits, these algorithms map a secret message onto a carrier without changing any aspect of the carrier. This technique not only allows you to use carriers that are not under your control, such as embedding a message at CNN's website, but they also allow you to be assured that current detection mechanisms will fail when analyzing your carrier.

During our analysis, we discovered several limitations in using immutable carriers for secret message transmission. First, these algorithms have a very limited bandwidth, so longer messages will require several different embeddings and transmissions, which could be flagged as anomalous. Secondly, the embedding techniques are probabilistic, so there is no guarantee that the secret message can be mapped onto a particular carrier. Lastly and perhaps most significant is the fact that the techniques all introduce a side channel through which the mapping information is transferred to the recipient, and this introduces the equivalent of the key exchange problem. Depending on the message that one wishes to embed and the size and versatility of the carrier, the resulting mappings can be quite large. Due to these limitations, we deduce that the VISA algorithms are a hybrid between encryption and steganography since the mapping onto the carrier in an innocuous fashion resembles steganography, but the transfer of the message, the structure of the mapping, and the potential attacks on the VISA algorithms all resemble those of encryption algorithms.

The first contribution of this paper is the design and implementation of a class of *variable interval* algorithms that we call VISA. We briefly present four algorithms that we implemented and tested on immutable carriers including an extended null-cipher algorithm for both natural language and images, a mapping using analytic functions across an image, and an algorithm for hashing image locations to create a mapping.

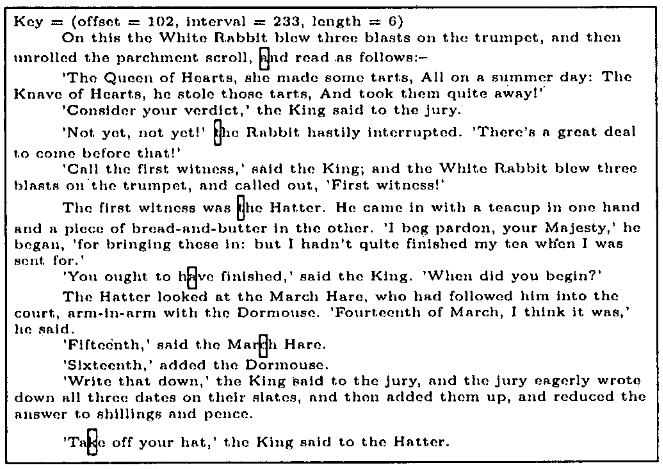
Second, we provide a thorough mathematical analysis of our proposed techniques. This analysis shows that immutable carriers have a greatly limited bandwidth compared to traditional steganographic algorithms. This limited bandwidth is an artifact of the VISA algorithm; we must search through and map our secret message to the carrier instead of simply modifying arbitrary bits of the carrier. This search may or may not be successful.

This paper is organized as follows. In §2, we introduce the variable interval algorithms for natural language and compare them to techniques that have been previously used. In §3, we discuss three different algorithms that we implemented for images, and in §4, we perform a mathematical analysis of our proposed techniques for text and images. Lastly, we conclude the paper in §5.

2. IMMUTABLE TEXT CARRIERS

During World War II, unencrypted transmissions were used to send embedded data. For example, the following message was sent by a German spy: [5, 7]

APPARENTLY NEUTRAL'S PROTEST IS THOROUGHLY DISCOUNTED AND IGNORED. ISMAN HARD


Key = (offset = 102, interval = 233, length = 6)

On this the White Rabbit blew three blasts on the trumpet, and then unrolled the parchment scroll, and read as follows:–

'The Queen of Hearts, she made some tarts, All on a summer day: The Knave of Hearts, he stole those tarts, And took them quite away!'

'Consider your verdict,' the King said to the jury.

'Not yet, not yet!' the Rabbit hastily interrupted. 'There's a great deal to come before that!'

'Call the first witness,' said the King; and the White Rabbit blew three blasts on the trumpet, and called out, 'First witness!'

The first witness was the Hatter. He came in with a teacup in one hand and a piece of bread-and-butter in the other. 'I beg pardon, your Majesty,' he began, 'for bringing these in: but I hadn't quite finished my tea when I was sent for.'

'You ought to have finished,' said the King. 'When did you begin?'

The Hatter looked at the March Hare, who had followed him into the court, arm-in-arm with the Dormouse. 'Fourteenth of March, I think it was,' he said.

'Fifteenth,' said the March Hare.

'Sixteenth,' added the Dormouse.

'Write that down,' the King said to the jury, and the jury eagerly wrote down all three dates on their slates, and then added them up, and reduced the answer to shillings and pence.

'Take off your hat,' the King said to the Hatter.

Figure 1: Embedded phrase 'attack' in an immutable text carrier

HIT. BLOCKADE ISSUE AFFECTS PRETEXT FOR EMBARGO ON BY-PRODUCTS, EJECTING SUETS AND VEGETABLE OILS.

Extracting the second letter in each word gives the following message:

PERSHING SAILS FROM NY JUNE 1.

This is an example of a *synthetic* carrier, where the text was generated to hide the message. This is in contrast to *mutable* carriers that are changed during the embedding process, and *immutable* carriers, which we discuss in this paper.

The null cipher algorithm used by this German spy has many limitations. Because it will only embed data in one letter of each word, the size of message that can be embedded is greatly limited. In a small corpus, matches are sparse. In addition to having few results, the shortest word found in your string limits the possible keys (i.e. in which letter position of the words you are embedding data). For example, if the string found in your carrier includes the word *the*, then the key is limited to the values 1, 2 or 3. Additionally, such a contrived message would never pass a bigram analysis [9] that analyzed the text for the probability that it was standard English. These analyses compare adjacent words for the probability that they would be seen next to each other, based on a huge amount of known English text.

If a match is found, transmitting a key containing the initial offset and interval suffices to transmit the message. Even if the key is intercepted in transit, there is little possibility of deriving the message without knowledge of, and access to, the corpus.

More data can be embedded in a carrier if the limitations of the naive implementation, namely that only one character can be used per word, are lifted. To do this, we have extended this algorithm to a *variable interval* algorithm. In this new algorithm, one of the embedded message's characters is hidden in every n letters. This algorithm would not be flagged by a bigram analysis as anomalous because we use corpora that contain actual spoken (or written) English words. As such, our technique is much less suspect. Figure 1 shows an example embedding using this algorithm on the message 'attack' in Lewis Carroll's *Alice In Wonderland* [11].

3. IMMUTABLE IMAGE CARRIERS

Images provide an obvious medium for practicing steganography with immutable carriers. By producing a map of locations within an image from which a message may be

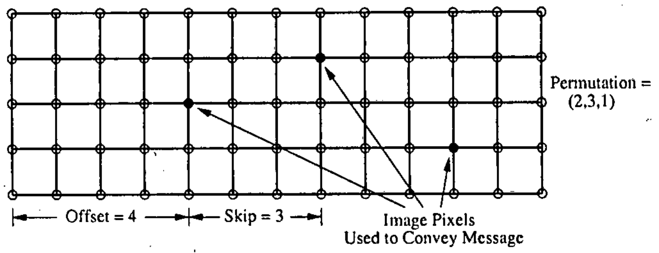


Figure 2: Choices of Image Pixels Using the Null Cipher Technique

extracted, we can effectively find our hidden message in a carrier without any modifications to that carrier. We explore three algorithms and implementations to accomplish this task in images in this section, and provide a mathematical analysis of these techniques in §4.

3.1 Null Ciphers to Map the Message

The null cipher technique for images is a variation of the text technique described in §2. The idea is to provide a (interval, offset) pair which will point out message data in an image. The only major variation from the text scheme is to first place the image into a two dimensional array, and then combine the interval and offset with a permutation on the heights of the image. Searching through different permutations of the heights has the advantage of searching a larger space in the corpus than if we were to search the corpus linearly. These permutations can be looked at as providing a variable-length interval in a single dimension. See Figure 2 for a pictorial view of this algorithm.

This scheme can be very costly. Since it involves searching the entire image, for a message of length n and a corpus of length m, this search is $O(m^n)$. For a message over a few characters in length, it becomes desirable to divide the message into smaller blocks to speed up the matching process, because of the exponential search time. This leads to decreased security, since breaking the message up requires more (offset,skip) values to be transmitted, increasing the ease of a statistical attack.

3.2 Hashing the Image

A second immutable-image algorithm that we implemented and analyzed hashed the image locations into bins, where the index of each bin is a byte of data from the image, and in each bin are locations where that byte can be found in the image. In order to map a byte of a secret message, we choose a random location from the bin indexed by that byte. This randomness is introduced to reduce the effectiveness of steganalysis, while at the same time provides an excellent probability of success.

While the previous schemes involved searching the image for a match of patterns, this hashing scheme only requires indexing an array using the byte to be matched, and choosing a random location in that bin. In terms of complexity, this scheme is $O(m)$ where m is the number of bytes in the image, and as such is much simpler than the matching process of the previous null cipher algorithm.

3.3 Analytic Functions to Map the Message

Similar to the techniques used in the previous sections, analytic functions provide a way to map message locations in the image data. We experimented with several trigonometric and polynomial functions. By varying the coefficients of these

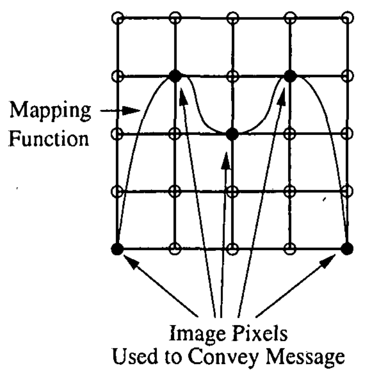


Figure 3: The Graph of an Analytic Function Which Maps Message Data onto Image Data

functions, the image is searched for matches with the message data.

Matching more bytes at a time increases the computational complexity exponentially, and reduces the probability of finding a match at all. Dividing a message into smaller blocks decreases the cost, but at the sake of weakened security, since dividing a message into smaller blocks produces more sets of coefficients, giving the steganalyst more information.

4. MATHEMATICAL ANALYSIS

In the simplest case, the encoding function f is a member of a specific class of functions, and we only need to find an appropriate set of parameters. We will explore one such class of functions, the "Offset-Interval" (VISA) class, in further detail. In the following equations, let L be the length, and L_x be the length of x. Let s be the secret message, c be the corpus, and o be the offset into that corpus. Lastly, let m be the maximum value allowed, and let i be the interval which we are discussing. This class is then defined as follows:

$$s \equiv f(c) = c_{i \cdot x + offset} \mid x \in \{0 \ldots (L-1)\} \qquad (1)$$

In this equation, c is the overt message and s is the secret message. Notationally, the subscript treats c as an array, that is, c_N means the Nth element in corpus. We arbitrarily define this element to be 8 bits of data so we can think of this as an ASCII character; we can just as easily use an element of another size. In this way, i is the distance between adjacent characters in s when reading through c, while o is the initial number of characters in c to ignore. L is simply the length of s in characters.

Thus, given a pre-selected c and a s we wish to encode, we need only find a triple {*offset, interval, length*}, such as the one we showed in Figure 1, that makes Equation 1 true (which may or may not be possible). If the class of functions has been pre-selected, this triple can then be thought of as a key to recovering s. Equivalently any arbitrarily chosen (not in a pre-selected class) and fully qualified function (without un-specified variables) f can be thought of as a key.

At this point, it should be obvious that this class is just the set of degree one polynomials over the non-negative integers. This was chosen for ease of encoding and decoding, and to be analogous to the original null cipher algorithm.

Given c, s, and the pre-selected class of functions given by Equation 1, what is the probability of successfully finding one or more "string encodings" — {*offset, interval, length*} triples that can extract s from c? We will derive an equation by calculating the maximal number of secrets that can be encoded in the corpus and the probability that one of these encoded strings actually matches our secret.

4.1 VISA Encoding Probability

Depending on the nature of the secret and corpus, and their respective lengths, it may or may not be possible to successfully encode s in c. Here we derive an equation for the probability of success. We begin with the following notation:

- L_d: The length of data d, where d is a corpus, etc.
- $N_d(b)$: The number of occurrences of byte b in d.
- $P_d(b)$: Probability of a randomly chosen byte in data d being b; $\frac{N_d(b)}{L_d}$

This section discusses the nature of encoded strings. In our case we are concerned with *secret*, which has length L_s and is made up of characters $S_1 S_2 \ldots S_{L_s}$ (Each character S_x is simply Equation 1 evaluated for a given x).

To start, the following bounds follow from our definitions and Equation 1, where e refers to the encoded secret.

$$\begin{aligned} 0 \leq &\quad o &\leq L_c - L_e \\ 1 \leq &\quad i &\leq \lfloor \frac{L_c}{L_s} \rfloor \\ 1 \leq &\quad L_s &\leq L_c \end{aligned}$$

But L_e, the length of the representation of s in terms of the amount of text it covers in c, is a function of i. This length is given as follows:

$$L_e = i \cdot (L_s - 1) + 1$$

This also allows us to find a tight upper bound on the maximal interval value. We know that the encoded secrets length must be less than or equal to the length of the corpus. So:

$$\begin{aligned} i \cdot (L_s - 1) + 1 &\leq L_c \\ i &\leq \frac{L_c - 1}{L_s - 1} \\ i_{max} &= \frac{L_c - 1}{L_s - 1} \end{aligned}$$

How many of these encoded secrets can we fit in our corpus? Think of this as a sliding window – We can place the encoded representation at the start of our corpus, or slide it over character by character until the end of our corpus. The number is simply $L_c - L_e + 1$.

So, how many different encodings E of s are possible in c? We sum the number of encoded secrets over all valid interval values, and substitute in the formula for L_e, canceling as we do:

$$E = \sum_{i=1}^{i_m} (L_c - i \cdot (L_s - 1))$$

We now evaluate this equation. First we remove constants to get:

$$E = L_c \left(\sum_{i=1}^{i_m} 1 \right) - (L_s - 1) \left(\sum_{i=1}^{i_m} i \right)$$

Evaluation of the sums gives:

$$E = L_c \cdot i_m - (L_s - 1) \cdot \frac{i_m \cdot (i_m + 1)}{2}$$

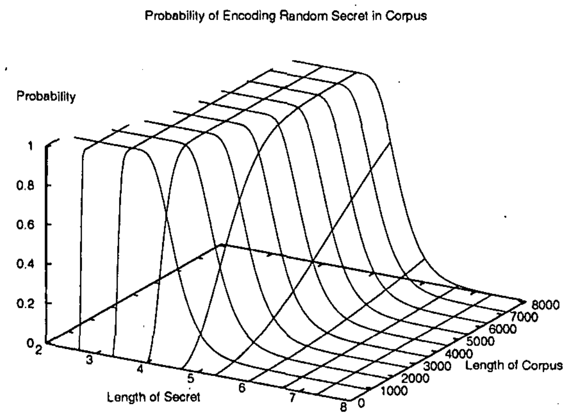


Figure 4: Best-case probability of finding an embedding

Substituting i_m gives:

$$E = L_c \cdot \frac{L_c - 1}{L_s - 1} - (L_s - 1) \cdot \frac{\frac{L_c-1}{L_s-1} \cdot \left(\frac{L_c-1}{L_s-1} + 1\right)}{2}$$

Which we reduce to:

$$E = \frac{L^2 - L_c \cdot L_s + L_c + L_s - 2}{2 \cdot (L_s - 1)} \tag{2}$$

Note that this is undefined with $L_s = 1$; this is a degenerate case. Our derivation implicitly assumes that there is more than one character in s; otherwise the probability of a successful encoding is just the probability that the character exists in the corpus, $P_c(character)$.

Now, if we knew the probability that one of these encodings matched our secret, we could calculate the probability of being able to encode s in c using this method. We know that the probability that a randomly selected set of characters in the corpus matches our secret is precisely:

$$P(s) = P_c(S_1) \cdot P_c(S_2) \cdot \cdots \cdot P_c(S_{L_s}) \tag{3}$$

To visualize how difficult it is to encode strings using this method, we graph a specific case of the equation. We assume that both the secret and the corpus are in the range A-Z and that each letter has an equal probability of $\frac{1}{26}$. Limiting ourselves to 26 characters instead of 255 obviously makes finding a match less difficult, and equalizing probabilities for each character maximizes $P(s)$ and hence the chances of a match. Figure 4 graphs the **best case**.

Figure 4 shows how simple it is to encode short strings (under 5 characters) in 8 kilobytes of c, but we have a cliff-like drop off after that. To emphasize the point, consider how long a corpus is required to make the probability of encoding a 16 character string over 50%. Solving the equation for L_c, one finds that on the order of 950 **gigabytes** of data required. For a probability of 90%, a corpus with 1.7 **terabytes** of data is required. The reader can easily calculate the probability of success given other circumstances. Recalling that this was the best case, the problem is obvious.

4.2 Generalization

Unfortunately, due to space constraints, we have only provided the analysis in the case where our encoding function f is a degree one polynomial. Any function that produces integers as output (locations in our corpus) could also be used. A general formula for the probability of successful encoding is given by Equation 4.

$$1 - (1 - probability\ of\ secret)^{number\ of\ encodings} \tag{4}$$

The perfect function would generate all possible substrings in the corpus (all $\sum_{x=1}^{L_c} \frac{L_c!}{(L_c-x)!}$ of them) given appropriate parameters.

5. CONCLUSION

In this paper, we presented a class of *variable interval* algorithms which we call *Variable Interval Symbol Aggregation (VISA)*. These algorithms work with immutable carriers which are not changed during the embedding process. Instead, they map a secret message onto a text or image carrier using a variety of techniques.

Our mathematical analysis of the VISA algorithms shows that the size of a potential secret message is relatively small, and the mapping information for these algorithms grows larger with each byte that is added to the secret message. However, this is to be expected with an algorithm that does not modify any of the bits of the carrier. Results from VISA algorithms are probabilistic and are not guaranteed. However, this is the trade-off that must be made for signature-free steganography. Addtionally, we noticed that the VISA algorithms had many characteristics in common with both steganography and encryption, and as such, we assert that the VISA algorithms are a hybrid between steganography and encryption.

6. REFERENCES

[1] R. J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, 16(4):474–481, May 1998. Special Issue on copyright and privacy protection.

[2] J. Fridrich, R. Du, and M. Long. Steganalysis of LSB encoding in color images. In *In Proceedings of the IEEE International Conference on Multimedia and Expo, August 2000.*, 2000.

[3] N. F. Johnson. Steganalysis of images created using current steganographic software. In *Proceedings of the Second Information Hiding Workshop*, April 1998.

[4] N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures.* Kluwer Academic Publishers, 2000.

[5] N. F. Johnson and S. Jajodia. Exploring steganography: Seeing the unseen. *IEEE Computer*, pages 26–34, February 1998.

[6] N. F. Johnson and S. Jajodia. Steganalysis: The investigation of hidden information. In *Proceedings of the 1998 IEEE Information Technology Conference*, September 1998.

[7] D. Kahn. *The Codebreakers - The Story of Secret Writing.* Scribner, New York, New York, USA, 1996.

[8] E. Kawaguchi and R. O. Eason. Principle and applications of BPCS steganography. In *Proceedings of SPIE's International Symposium on Voice, Video, and Data Communications*, November 1998.

[9] S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering, 1995.

[10] Outguess. http://www.outguess.org/.

[11] Project Gutenberg. http://promo.net/pg.

[12] N. Provos and P. Honeyman. Detecting steganographic content on the internet. In *In Proceedings of the Network and Distributed Systems Security Symposium (NDSS) 2002*, 2002.